|  |  |  |
| --- | --- | --- |
| **AI2B** | Hubkiewicz Jakub  Album 46525  Grupa 2 | **Wersja 1** |
|  |
| **LAB C** |

Routing i Serwisy w Angular

# Spis treści

[Spis treści 1](#_Toc151039756)

[Cel zajęć 1](#_Toc151039757)

[Rozpoczęcie 1](#_Toc151039758)

[Uwaga 2](#_Toc151039759)

[Inicjalizacja projektu 2](#_Toc151039760)

[Implementacja serwisu przechowującego dane w LocalStorage 5](#_Toc151039761)

[Implementacja listy danych 7](#_Toc151039762)

[Dodawanie komponentów i serwisu 11](#_Toc151039763)

[Implementacja serwisu RandomService 12](#_Toc151039764)

[Implementacja komponentu Random 14](#_Toc151039765)

[Implementacja komponentu listComponent 17](#_Toc151039766)

[Commit projektu do GIT 19](#_Toc151039767)

[Podsumowanie 19](#_Toc151039768)

# Cel zajęć

Celem głównym zajęć jest zdobycie umiejętności:

* przetwarzania danych z zapisem w LocalStorage i z wykorzystaniem serwisów,
* obsługa routingu w Angular.

W praktycznym wymiarze stworzony zostanie prosty projekt pozwalający na dodawanie danych osobowych i zarządzanie nimi w LocalStorage przeglądarki internetowej.

# Rozpoczęcie

Rozpoczęcie zajęć. Powtórzenie tworzenia komponentów, serwisów. Powtórzenie wiązań.

Wejściówka?

# Uwaga

Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

# Inicjalizacja projektu

Wejdź terminalem do katalogu C:\...\Desktop\ai2b i zainicjalizuj projekt z wykorzystaniem komendy:

> ng new lab-c

Standardowo kreator zapyta o konfigurację routingu (wybrać **Tak**) oraz preprocesor CSS (zostawić zwykły CSS). Zainicjalizowane zostanie także repozytorium GIT.

Po zakończonej instalacji, uruchom aplikację w trybie deweloperskim z wykorzystaniem komendy:

> cd C:\...\Desktop\ai2b\lab-c

> ng serve --port=00000

Tradycyjnie, do pliku src/styles.css dodaj znak wodny ze swoim numerem albumu:

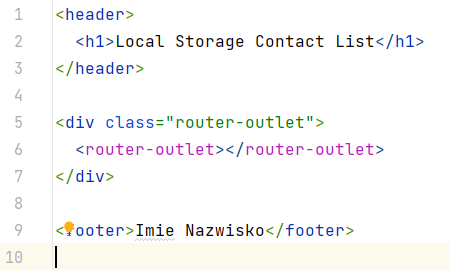
body {

background: url("https://placehold.co/100x100/FFFFFF/EFEFEF/png?text=00000");

}

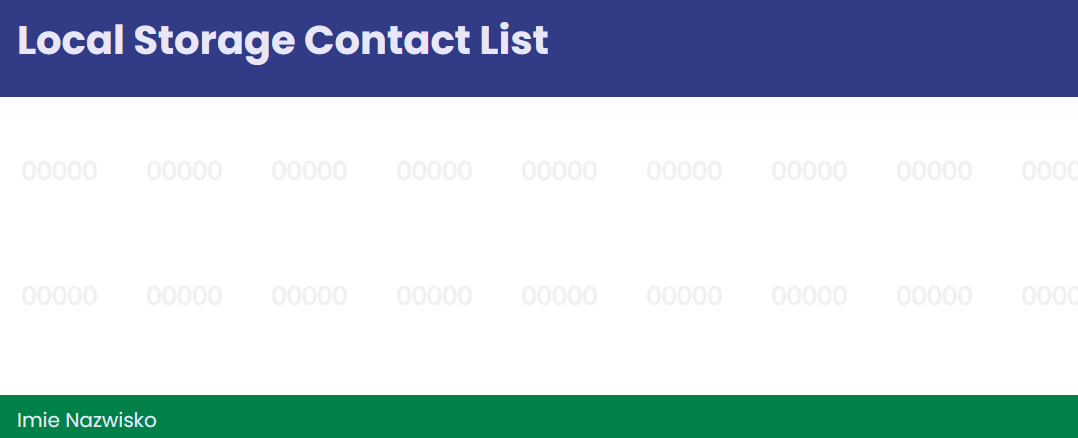
Uruchom przeglądarkę pod adresem: http://localhost:00000.

Edytuj zawartość pliku src/app/app.component.html. Usuń domyślnie wygenerowaną zawartość. Zostaw jedynie znacznik <router-outlet>. Utwórz własne, stałe elementy interfejsu aplikacji, np. nagłówek i stopkę. Np.:



Dodaj style w plikach src/styles.css (globalne) i src/app/app.component.css wg własnego uznania.

Przykładowy efekt:



Następnie dodaj komponenty, pod przyszłe widoki:

* ListComponent – wyświetlanie listy osób
* DetailsComponent – szczegóły wybranej osoby
* AddPersonComponent – dodawanie danych nowej osoby
* NotFoundComponent – informacja o błędnej ścieżce routingu.

W tym celu otwórz osobny terminal i wykonaj polecenia:

> ng generate component list --skip-tests

> ng generate component details --skip-tests

> ng generate component addPerson --skip-tests

> ng generate component notFound --skip-tests

Utworzone zostaną 4 katalogi komponentów.

W automatycznie wygenerowanym module routingu (plik src/app/app-routing.module.ts) ustaw ścieżki routingu dodając elementy w tablicy `routes`:

* list – powinno kierować do komponentu ListComponent
* details/:id – powinno kierować do komponentu DetailsComponent, ścieżka zawiera parametr
* add – powinno kierować do komponentu AddPersonComponent

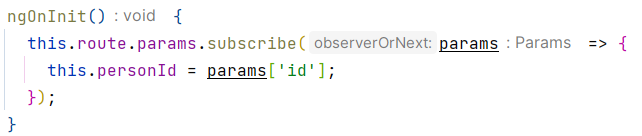
Dodatkowo należy ustawić domyślne przekierowanie ze ścieżki ”” (pusta) na ”/list” oraz ze ścieżki ”\*\*” (dowolna inna) na komponent NotFoundComponent.

Ponadto na widoku komponentu DetailsComponent należy wyświetlić bieżące ID (patrz. Lab A oraz wykłady).

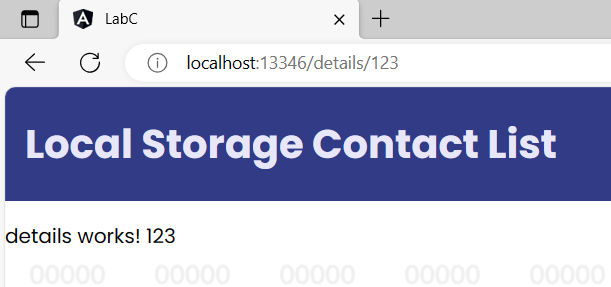
Przykładowa implementacja routingu:



Przykładowa subskrypcja parametru ID w DetailsComponent:



Przykładowy wygląd strony /details/123:



W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/f6902ba9fd002753586a92e61f5575ea43f39aed>

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę /list. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz domyślny szablon „list works!”

Obraz zawierający oprogramowanie, tekst, Oprogramowanie multimedialne, Ikona komputerowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę /details/42. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz wyświetlona w komponencie DetailsComponent wartość 42 (z URLa).

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, Ikona komputerowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę /add. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz domyślny szablon „add-person works!”

Obraz zawierający oprogramowanie, tekst, Ikona komputerowa, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę /not-found (lub dowolny inny nieistniejący). Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz domyślny szablon „not-found works!”

Obraz zawierający oprogramowanie, tekst, Oprogramowanie multimedialne, Ikona komputerowa

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja serwisu przechowującego dane w LocalStorage

Utwórz interfejs reprezentujący dane osobowe przetwarzane w aplikacji komendą:

> ng generate interface person

Utworzony zostanie plik src/app/person.ts. Uzupełnij jego zawartość, przykładowo:



Następnie użyj terminala, żeby wygenerować serwis PersonLsService:

> ng generate service personLs --skip-tests

Utworzony zostanie plik src/app/person-ls.service.ts. Podmień jego zawartość na poniższą:

import { Injectable } from '@angular/core';

import {Person} from "./person";

@Injectable({

providedIn: 'root'

})

export class PersonLsService {

readonly KEY = 'stored-people-data';

constructor() { }

// public getAll(): Person[] {

// // set response people variable to an empty array

// // get data from localstorage

// // if anything was fetched, parse using JSON.parse() and assign to people

//

// // return all people

// }

//

// public getPerson(index: number): Person {

// // get all people and return the one at [index] position

// }

//

// public addPerson(person: Person): void {

// // get all people

// // push person to the array of people

// // update localstorage with the array contents serialized using JSON.stringify()

// }

//

// public deletePerson(index: number): void {

// // get all people

// // use splice() to remove person at [index] position

// // update localstorage with the new values of people array

// }

}

W powyższym kodzie przygotowano sygnatury 4 metod:

* do pobierania listy wszystkich osób z local storage,
* do pobierania osoby pod określonym indeksem w tablicy wszystkich osób
* do dodawania nowej osoby
* do kasowania osoby pod zadanym indeksem.

Odkomentuj te funkcje i wypełnij ich definicje. W przypadku problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/6ded9bb153ee349e74f28c595cf2ea3ea6211864#diff-2f7e27c391409db46653869eef8522a755ecad6e46789b4cd8223b9be5551936>

Umieść poniżej zrzut ekranu kodu metody getAll():

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu metody getPerson():

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu metody addPerson():

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu metody deletePerson():

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja listy danych

Korzystając z narzędzi deweloperskich aplikacji (karta Application -> Local Storage) ustaw przykładowe dane do pobrania przez listę (klucz zgodny z tym określonym w serwisie PersonLsService):

[

{

"firstName": "Alice",

"lastName": "Alyska",

"age": 21,

"address": {

"city": "Szczecin",

"street": "Zolnierska 49",

"postCode": "71-210"

}

},

{

"firstName": "Bob",

"lastName": "Bobsky",

"age": 22,

"address": {

"city": "Szczecin",

"street": "Zolnierska 49",

"postCode": "71-210"

}

}

]

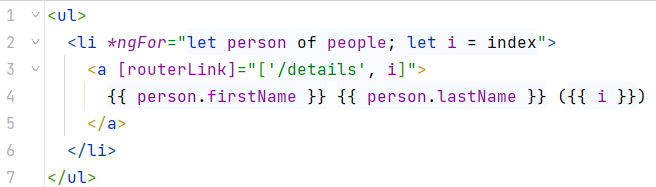
Następnie zaimplementuj wyświetlanie listy osób. Dla każdej osoby wyświetlaj imię, nazwisko oraz identyfikator na liście. Każda pozycja musi być jednocześnie łączem do ekranu szczegółów danej osoby, przykładowo:



Podczas implementacji, w komponencie ListComponent należy zainicjalizować tablicę elementów typu Person[], wstrzyknąć serwis i w metodzie ngOnInit() wczytać za jego pomocą listę wszystkich osób (metoda getAll()).

Na widoku należy użyć dyrektywy \*ngFor, żeby wyświetlić element listy dla każdej osoby z pozyskanej tablicy osób.

Przykładowa implementacja:

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/6ded9bb153ee349e74f28c595cf2ea3ea6211864>

Umieść poniżej zrzut ekranu przeglądarki z wyświetloną listą osób:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/list/list.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, Czcionka

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/list/list.component.html:

Obraz zawierający zrzut ekranu, tekst, Oprogramowanie multimedialne, oprogramowanie

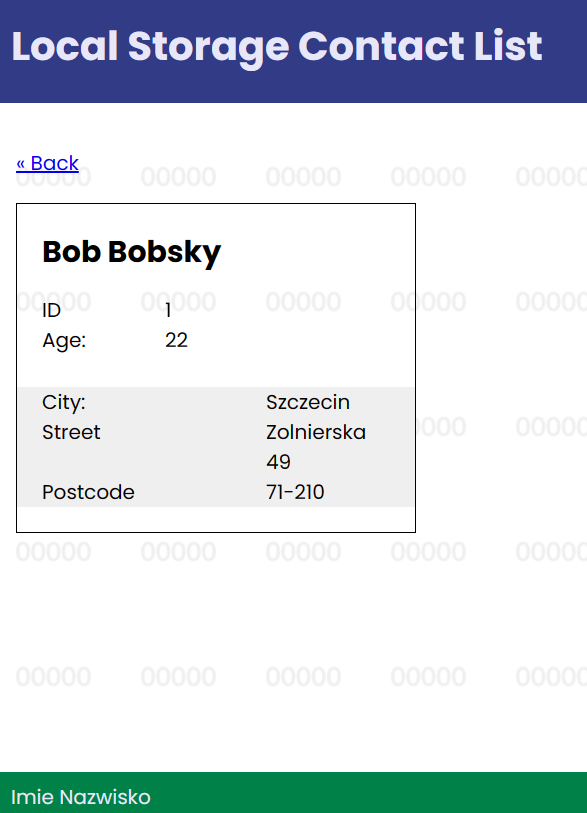
Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja komponentu szczegółów

W tej sekcji zaimplementujemy komponent DetailsComponent. Wstrzyknij serwis i wczytuj z local storage pojedynczy element Person wskazany wartością przekazaną w ścieżce routingu. W widoku dokonaj interpolacji uzyskanych danych, stylując wedle uznania.

Przykładowa implementacja:

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/56736f14c7baf280d04bf5f4a72e510a192dada2>

Umieść poniżej zrzut ekranu przeglądarki z wyświetlonymi szczegółami jednej osoby:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/details/details.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/details/details.component.html:

Obraz zawierający tekst, elektronika, zrzut ekranu, wyświetlacz

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/details/details.component.css:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja dodawania danych

Gdzieś na stronie umieść link nawigujący do widoku dodania nowego elementu danych /add.

W komponencie AddPersonComponent zdefiniuj pole typu Person reprezentujące nowe dane osobowe i zainicjalizuj je. Przykład inicjalizacji:

person: Person = {

address: {}

};

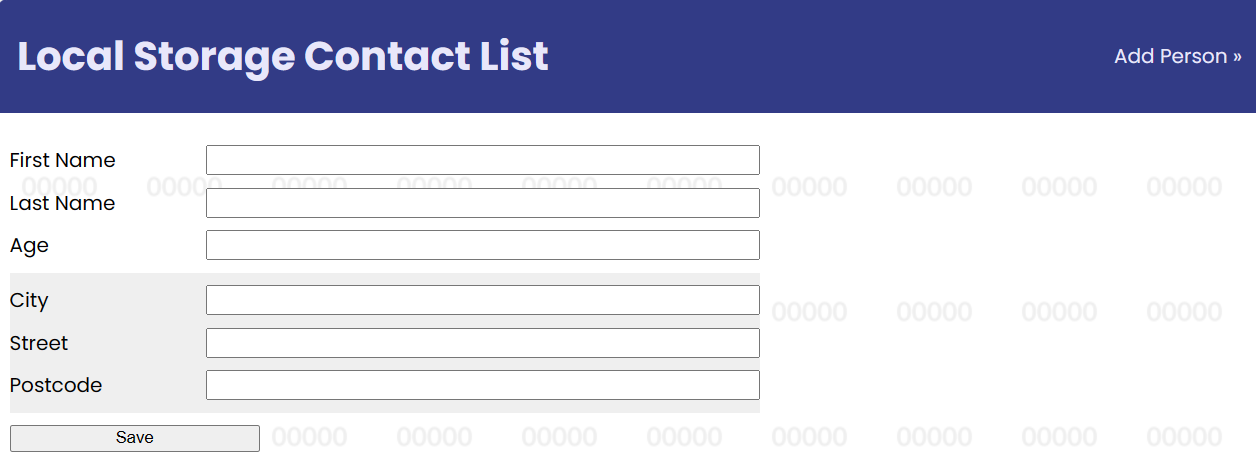
Wstrzyknij opracowany wcześniej serwis oraz obiekt typu Router potrzebny do przełączenia widoku po pomyślnym zapisie danych. Dodaj metodę save(), która po wywołaniu zapisze dane osobowe z pola person do local storage za pomocą metody addPerson() serwisu, a następnie przekieruje użytkownika na listę.

Mając skończoną implementację części komponentowej, w widoku src/app/add-person/add-person.component.html zbuduj formularz do uzupełnienia danych osoby i jej adresu. Pola formularza powiąż dwukierunkowo z polami obiektu person za pomocą [(ngModel)], np. [(ngModel)]="person.firstName".

Dodaj także przycisk zapisywania, który wykona metodę save() komponentu.

Pamiętaj o zaimportowaniu brakujących modułów (w tym FormsModule) w src/app/app.module.ts.

Przykładowa implementacja:



W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/f71256ddb3f97dc16d16b2e2402d8aed3e0f01a9>

Umieść poniżej zrzut ekranu przeglądarki z ekranem dodawania osoby:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu listy przed dodaniem osoby:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu listy po dodaniu osoby:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/add-person/add-person.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/add-person/add-person.component.html:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie Obraz zawierający zrzut ekranu, tekst, Oprogramowanie multimedialne, oprogramowanie

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/add-person/add-person.component.css:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

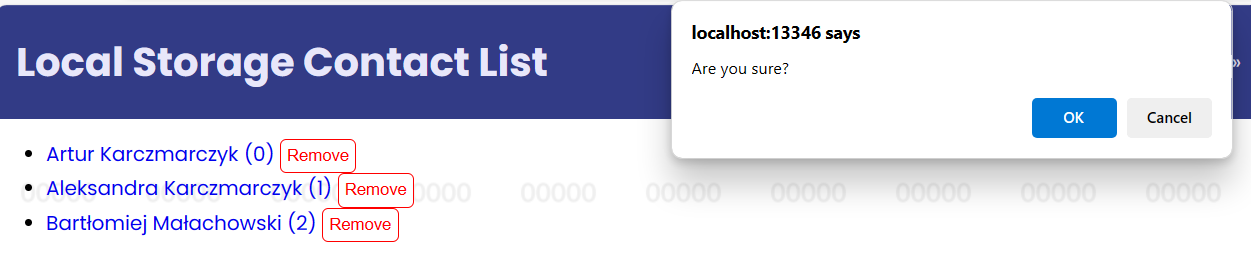
Opis wygenerowany automatycznie

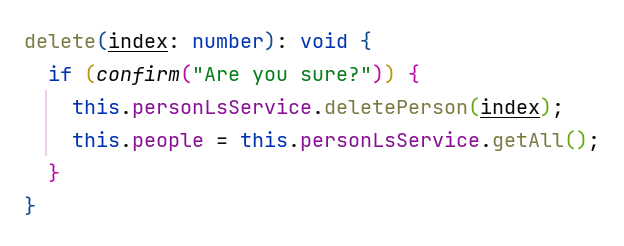
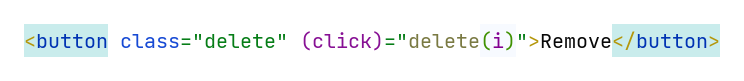
|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja usuwania danych

W komponencie ListComponent dodaj metodę akcji delete(index: number) usuwającą poprzez serwis local storage element wskazany wartością parametru index. W widoku dodaj dla każdego elementu wyświetlanego na liście link lub przycisk dowiązujący zdarzenie (click) do wywołania metody delete() z parametrem będącym indeksem tego elementu w tablicy danych.

Przykładowa implementacja:





W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/8af49aff01884efafd9dd86d6135362edb31a0a2>

Umieść poniżej zrzut ekranu przeglądarki z ekranem listy osób przed usunięciem osoby:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu listy po usunięciu osoby:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/list/list.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/list/list.component.html:

Obraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, oprogramowanie

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/list/list.component.css:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

W międzyczasie były zmiany

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Commit projektu do GIT

Utwórz repozytorium publiczne GitHub na tę część kursu. Wyślij swój projekt do repozytorium (push). Upewnij się, czy wszystko dobrze się wysłało. Jeśli tak, to z poziomu przeglądarki utwórz branch o nazwie lab-c na podstawie bieżącej gałęzi kodu.

W zależności od przyjętej konwencji (jedno repo per laboratorium kontra jedno repo na wszystkie laboratoria), konieczne może być usunięcie katalogu .git i ponowna samodzielna inicjalizacja.

Podaj link do brancha lab-c w swoim repozytorium:

<https://github.com/huuuuubi/AI2B-L-grN2-Hubkiewicz-Jakub/tree/lab-c>

# Podsumowanie

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Nie ukrywam – to było ciężkie zadanie. Już od początku miałem problemy z wyświetlaniem czegokolwiek, a potem kod nie działał i się nie pokrywał z tym z repozytorium. Szczerze muszę przyznać, że musiałem się mocno posiłkować załączonymi pomocami naukowymi.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.